コンパイラ実験課題
第一回

05-191022 平田 賢吾

2019 年 10 月 9 日

問 1

説明

syntax.t と kNormal.t に対応した print 関数をそれぞれ syntax.ml と kNormal.ml に作成した. syntax.print はスライドに近い英大文字で出力するように, kNormal.print は ocaml の構文に近い形で出力するようにした.

それぞれ make を実行すると, typing.f の冒頭, kNormal.f の最後に print 命令が置かれているため, 標準出力に出力されるようになっている.

下の実行例は mincaml/test/test-for-compiler.ml においてある.

実行例

```
----- min-caml code -----
let rec f x =
  let rec g y =
    x + 2 - y
  in
  g
in
print_int (3 + ((f 4) 9))

----- syntax.print -----
LETREC f (VAR of )
  (x : VAR of )
  LETREC g (VAR of )
    (y : VAR of )
    SUB
      ADD
        VAR x
        INT 2
      VAR y
    VAR g
```

```
  APP
    VAR print_int
    ADD
      INT 3
      APP
        APP
          VAR f
          INT 4
        INT 9

----- KNormal.print -----
letrec f : (INT -> (INT -> INT)) =
variables : (x : INT)
  letrec g : (INT -> INT) =
  variables : (y : INT)
    let Ti8 : INT =
      let Ti7 : INT =
        int 2
      in
      add x Ti7
    in
    sub Ti8 y
  in
    g
in
  let Ti6 : INT =
    let Ti1 : INT =
      int 3
    in
    let Ti5 : INT =
      let Tf3 : (INT -> INT) =
        let Ti2 : INT =
          int 4
        in
        app
          f
          Ti2
      in
      let Ti4 : INT =
        int 9
      in
      app
        Tf3
        Ti4
    in
    add Ti1 Ti5
  in
  extfunapp
    print_int
```

Ti6

考察

最初に syntax.print を実装してみたところ, なんだか慣れない見た目をしており少々読みにくく感じたため kNormal.print ocaml っぽく実装した. kNortmal.t は最初のコードより文字が増えているから ocaml っぽくすることで in で改行したりしたのがさらにコードの量を増やして少々読みにくい感じを与えるものの, 個人的にはこっちの方がまあ読みやすいかなと思う.

問 2

説明

lexer, parser でエラーした際は元コードの何行何番目の文字が間違っているかを表示した. 型エラーを発見した時は型エラーを起こした term を下線付きで出力し,

”This expression has type HOGE but an expression was expected of type FUGA.”

という形式で出力するようにした.

下の実行例 3 つはは mincaml/test/test-for-compilerN.ml (N = 2, 3, 4) においてある.

実行例

```
  1
  2
  3
  4
  5
  6
  7 let
  8 * = 0 in print_int 0
Fatal error: exception Failure("␣line␣8:0-1␣Error::␣unknown␣token␣*␣")

  1
  2
  3 let 0 =
  4 0
  5 in print_int 0
Fatal error: exception Failure("␣line␣3:4-5␣Parse␣Error:")

  1
  2
  3
  4
  5 print_int
  6 (0 +
  7 0.0)
```

```
ADD
  INT 0
  FLOAT 0.
Error: This expression has type FLOAT but an expression was expected of type INT.
```

考察

基本的に ocaml 本家と似たエラーメッセージを目指した. 型エラーのメッセージは何行目か表示してないが, コードの量が多い時はあったほうがいかもしれない.